HIToolbox: An Architectural Overview

Session 204

HIToolbox: An Architectural Overview

Xavier Legros
Mac OS X Evangelist

Welcome to HIToolbox Day

HIObject

HIView

New Controls

HIToolbox: An Architectural Overview

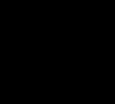
Ed Voas
HIToolbox Manager

What You'll Learn

- Past, Present, and Future
- HIToolbox directions
- New foundation for the future
    - Why is it important?

What Is HIToolbox?

- Carbon User Interface environment
  - Appearance
  - Events
  - Menus, Windows, Controls, etc.
- Also called “High Level Toolbox”



HIToolbox Charter

- Accelerate App Development
- More standard functionality
- Object-oriented approach
- Easier to customize to your needs
- Make it fun to write a Mac OS app

History

Mac OS 8.0

- Platinum Appearance
- Many new controls
- Keyboard focus
- Control embedding
- Live Scrolling

But that wasn’t good enough . . .

Mac OS 8.5

- Full-blown Appearance Manager
- Native High Level Toolbox
  - Foundation for our Mac OS X Toolbox
- Floating windows
- Window proxies
- Properties

But it still wasn't good enough . . .

Mac OS 9.0

- Control drag and drop
- Lots of other stuff
    - Well, who are we kidding?
    - We were busy . . .

Mac OS X 10.0

- Ported native Toolbox to Mach/BSD
- Data structure opacity (this was a huge plus)
- Carbon Events
- Aqua
- Window Buffering
- Window Groups

But even that wasn't good enough . . .

Mac OS X 10.1

- Many performance improvements
- Services Menu Integration
- A boatload of internal changes

But That's Not Good Enough!

Not Enough?

- Still hard to override default behavior
- Hard to write custom controls
- Different Def Proc model per manager
- Incomplete widget set

Jaguar HIToolbox

- New view architecture
- Accessibility APIs
- New controls: Combo Box, Toolbar
- Drawers
- Grand unification of Toolbox objects
    - Reduce and simplify

From Toolbox to Framework

- Embedding, new controls
- Carbon Events
- New view architecture
- More in the future

What Will the New HIToolbox Bring You?

A New View System

HIView: Composited Views

- Composited, one-pass drawing
- CG drawing is native
- Floating-point coordinate system
    - Though do not go subpixel-happy
- Overlapping views

Accessibility

Accessibility

- Allows third-party applications to peek inside your application to enable accessible solutions
- High-level actions: press, select, etc.
- APIs and Carbon Events to allow your application to be accessible as well

New Elements

New Elements

- Combo Box
- Toolbar
- Drawers

Demo

A New Model to Unify

Pre-Jaguar

- Each Toolbox Manager had a different implementation (though similar)
  - Properties
  - DefProc model (WDEF, CDEF, MDEF)
  - Retain/Release

Jaguar: A Unified Model

- Common base-class for common objects
- One implementation of common things
- Polymorphic functions
- Birth of HIObject

What Is HIObject?

- A new common base class for Toolbox objects
  - Menus, Windows, Controls, Toolbars
- Provides mechanism to subclass
- HIObject is actually a CF-based type
  - CFRetain/Release
  - CFArray/CFDictionary

HIObject: It's an Object Model

- HIObject is the data store
- Carbon Events are the methods
- All HIObjects are implicitly Event Targets

Advantages for Developers

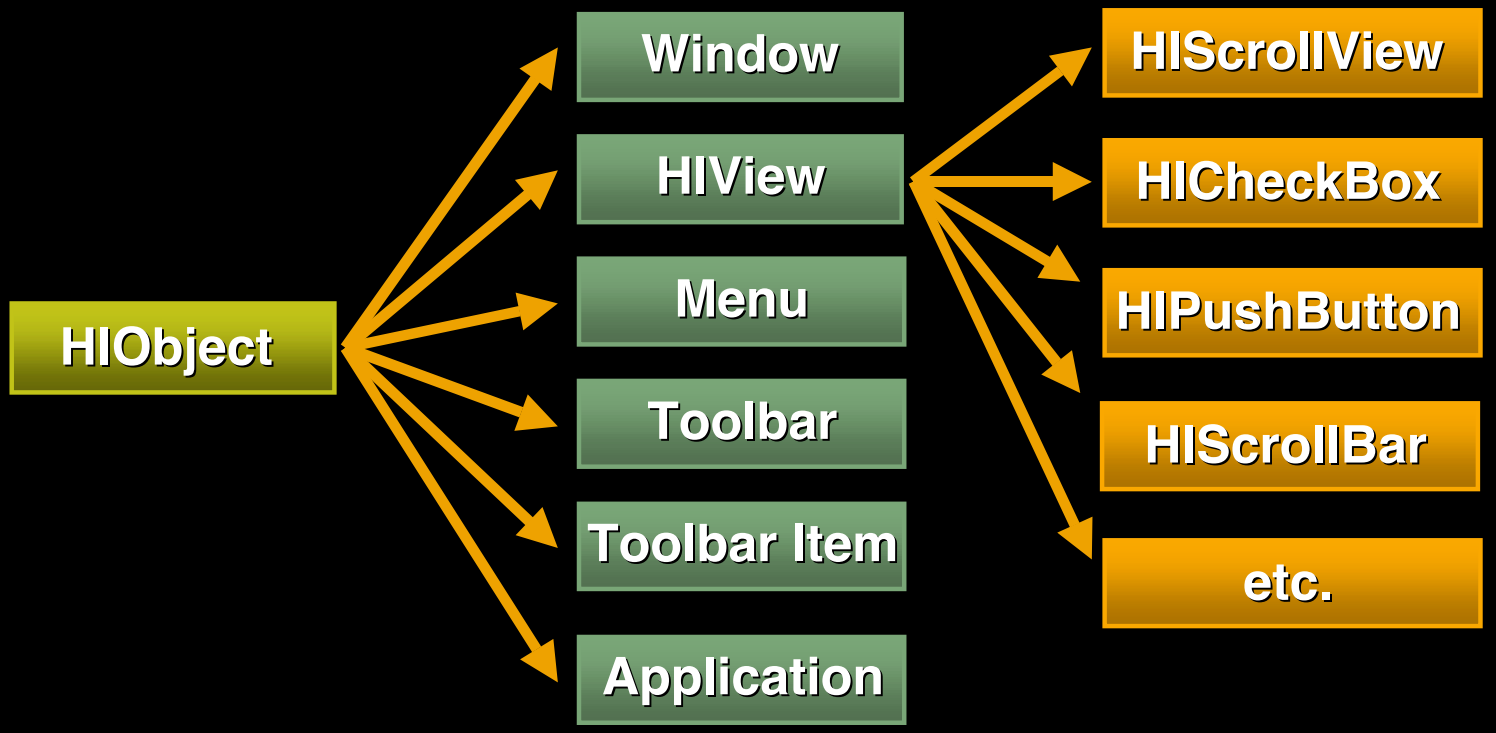
- Consistency (API and usage)
  - Polymorphic functions
- Easy to implement custom objects
- Easy to derive from standard objects
- Does not impact your current usage
- Ability to create your own event targets

Toolbox Objects Are HIObjects!

- Here's proof:

```
typedef HIObjectRef ControlRef;
typedef HIObjectRef HIToolbarRef;
typedef HIObjectRef MenuRef;
typedef HIObjectRef WindowRef;
```

Current Hierarchy



Examples of Usage

- Custom toolbar item
- Custom views
- Custom event targets

Introduction to HIObjects

- How to register subclasses and create them
- HIObject construction process
- Instance data and how it relates to HIObject
- Dynamic casts
- Polymorphic functions

Creating Custom Objects

- First, register a subclass . . .

```
HIObjectRegisterSubclass(
    CFStringRef                 inBaseClass,
    CFStringRef                 inYourClass,
    OptionBits                  inOptions,
    EventHandlerUPP             inEventHandler,
    UInt32                      inNumEvents,
    const EventTypeSpec *       inEventList,
    void *                      inConstructData,
    HIObjectClassRef *          outClassRef );
```

Creating Custom Objects

- . . Then call HIObjectCreate:

```
OSStatus HIObjectCreate(
    CFStringRef             inClassID,
    EventRef                inInitEvent,
    HIObjectRef*            outObjectRef );
```

Example HIToolbarItem

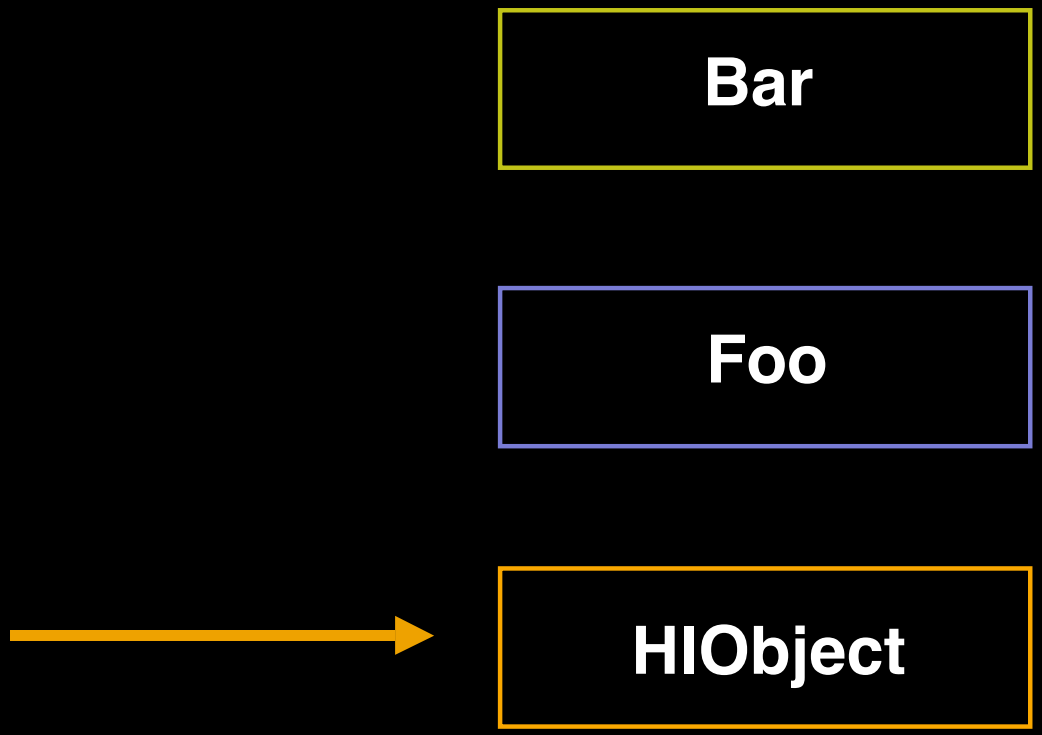
```
#define kMyCustomItemID
    CFSTR( “com.myco.customitem” )

HIObjectRegisterSubclass(
     kMyCustomItemID,
    kHIToolbarItemClassID,
    0, // no options
    MyClassHandler,
    GetEventTypeCount( kCustomEvents ),
     kCustomEvents,
    0, // no construct data
    &myClassRef );
```

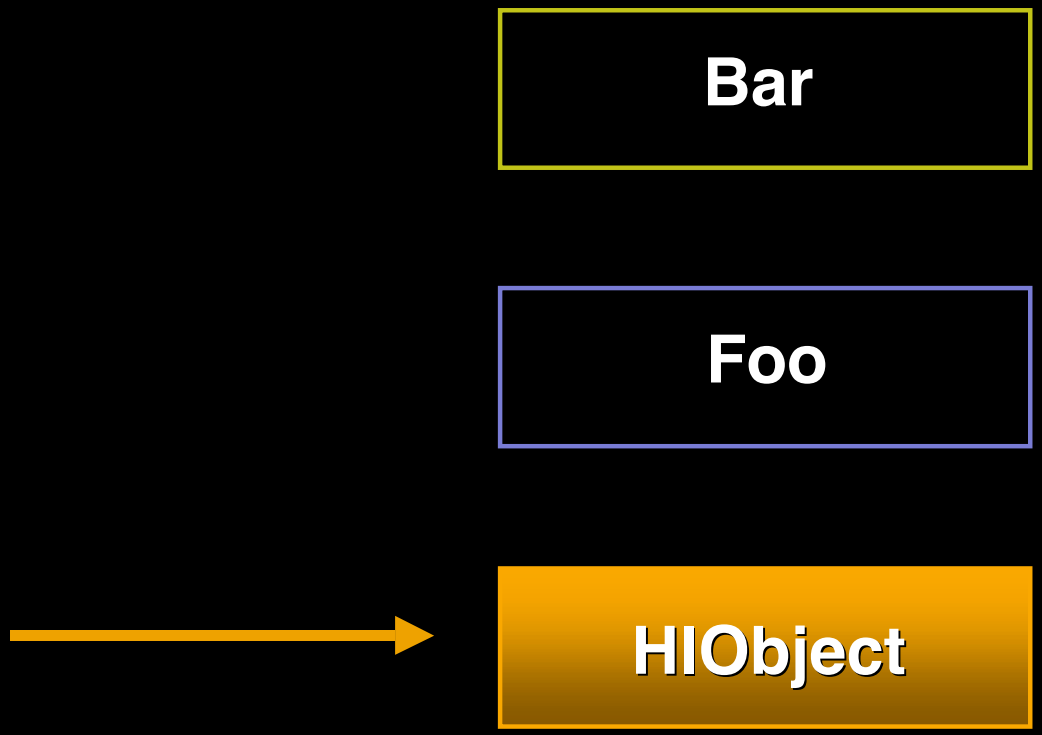
HIObject Construction

- Bottom-Up construction
- Two-phase: construct and initialize
- Three important events
    - kEventHIObjectConstruct
    - kEventHIObjectInitialize
    - kEventHIObjectDestruct
- Construct and Destruct are required!

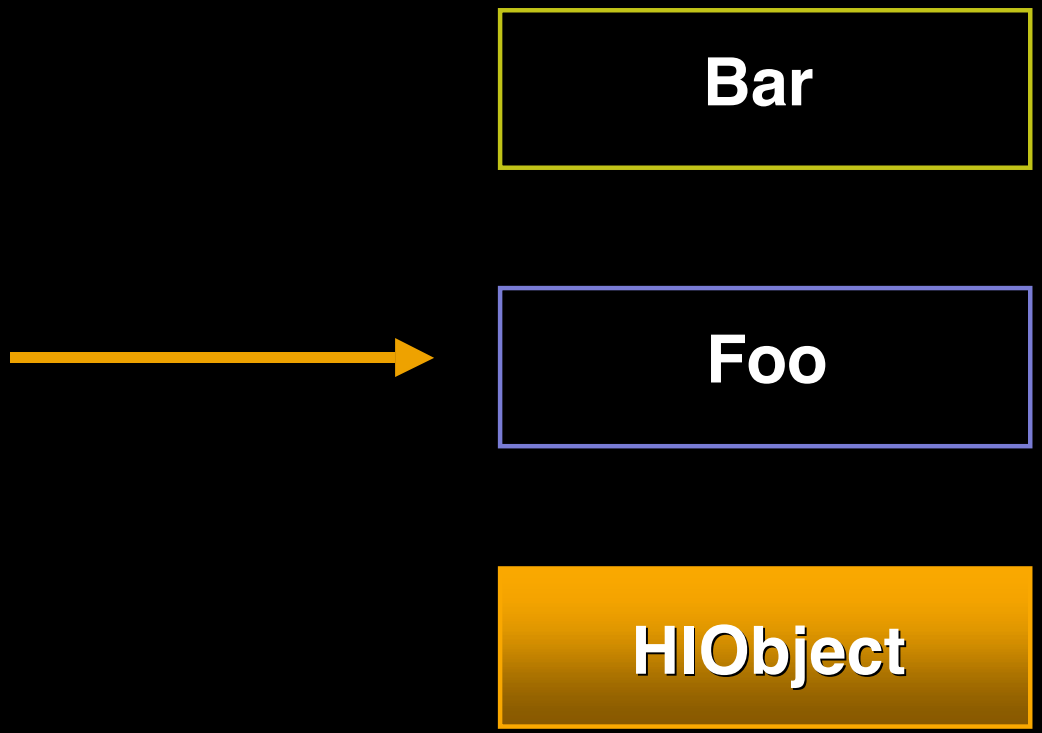
Construction Order



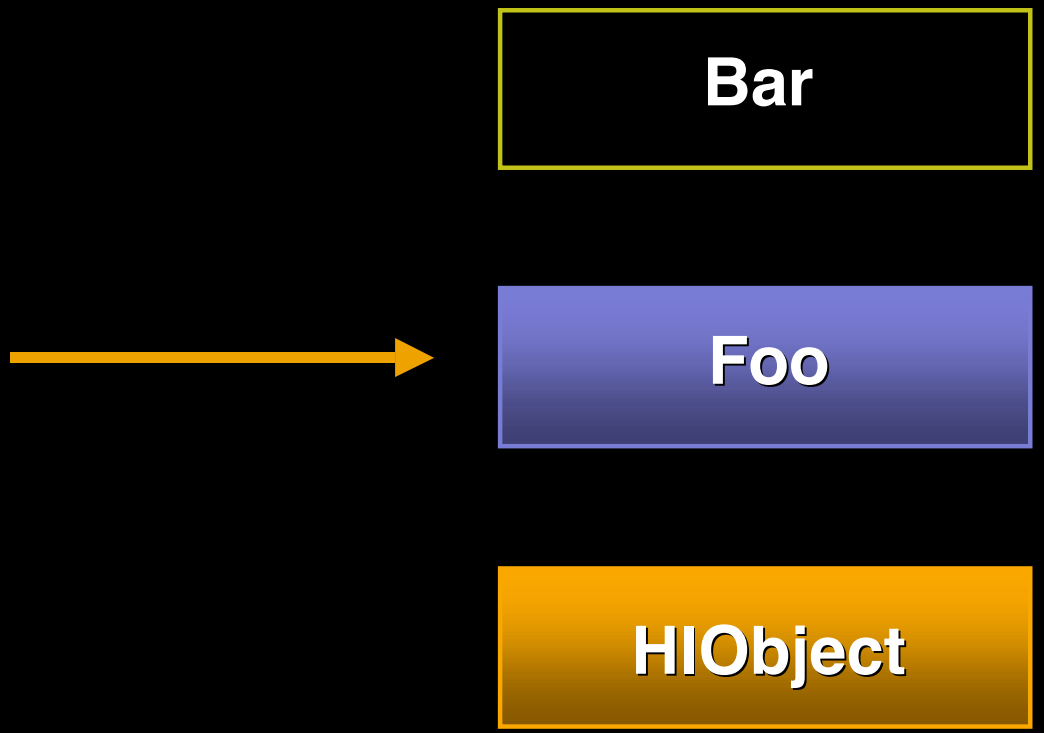
Construction Order



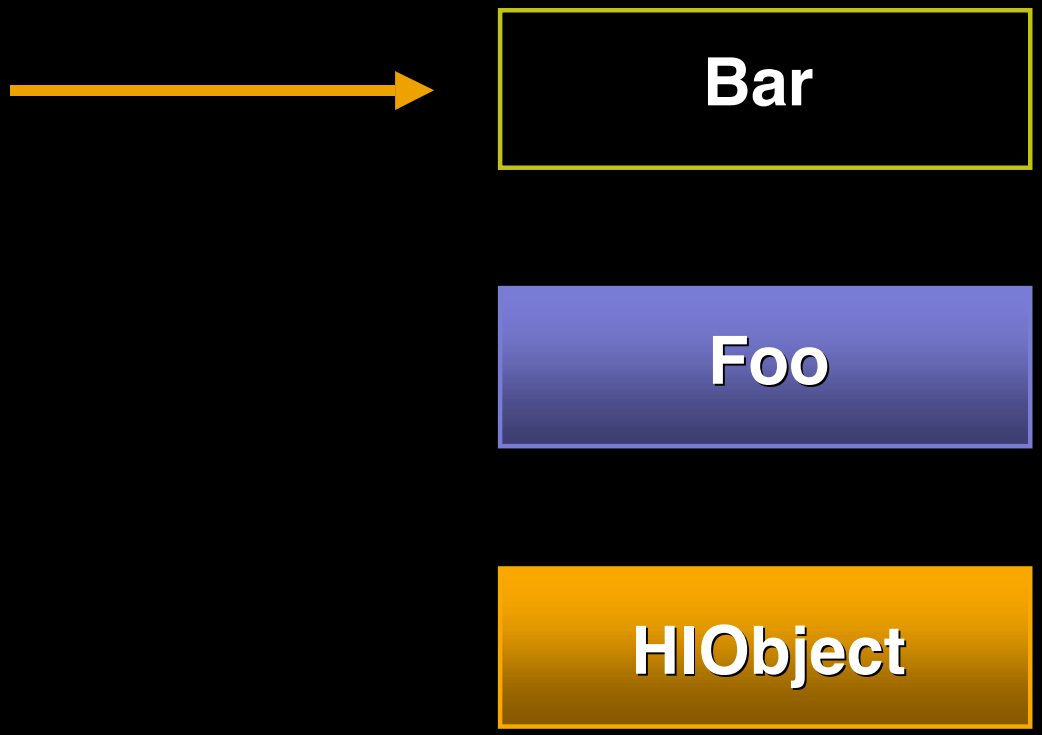
Construction Order



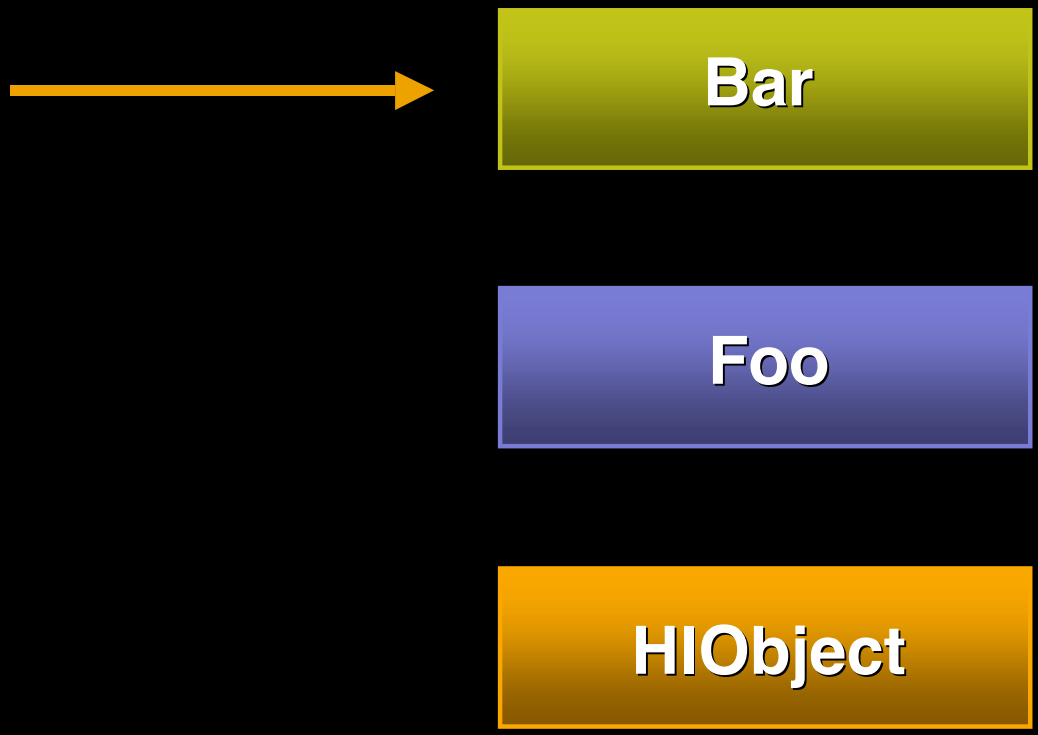
Construction Order



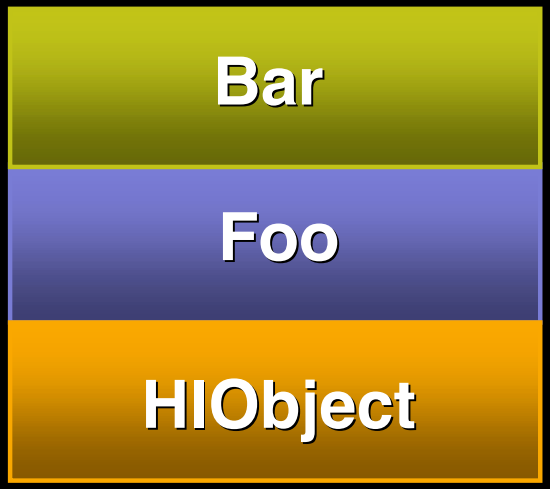
Construction Order



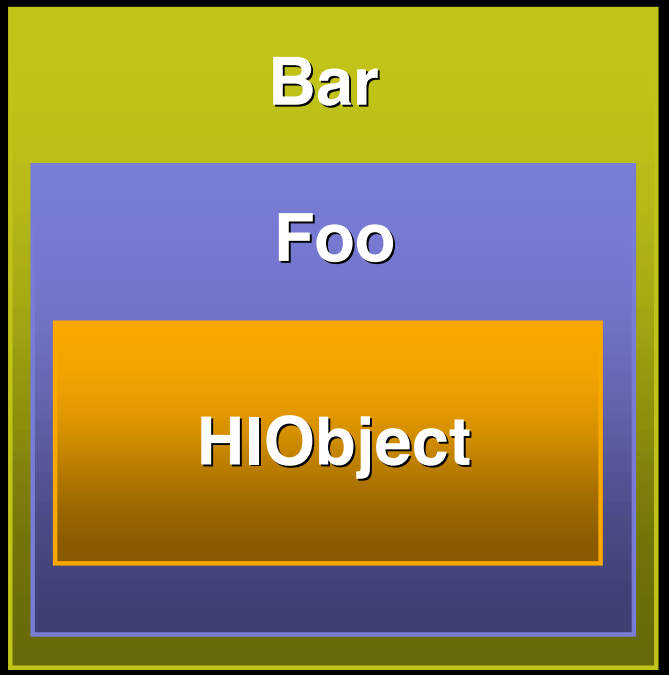
Construction Order



Construction Order



Construction Order



Construct Event

- Event handler is called directly
  - Cannot use CallNextEventHandler
  - Passed data specified when registering
- Your subclass should create instance data to hold custom information
  - Set in Carbon Event as return parameter

Instance Data

- Can be whatever you want
- Must be typeVoidPtr on return
- Used in two ways
    - Stored with classID for dynamic casting
    - Set into event handler as user data

Instance Data vs. HIObjectRef

- An HIObjectRef is not your instance data
  - This is unlike C++
- How do you get your data back?
  - HIObjectDynamicCast

HIObjectDynamicCast

```
void MyFunRoutine( HIObjectRef inMyObject )
{
   MyObjecPtr*        ptr;

   ptr = HIObjectDynamicCast(    kMyClassID,
                                 inMyObject );

   if ( ptr )
   {
         // here we know it's one of our objects
   }
}
```

Handling the Construct Event

```
OSStatus
MyEventHandler(
      EventHandlerCallRef          inCallRef,
      EventRef                     inEvent,
      void*                        inUserData )
{
   switch ( GetEventClass( inEvent ) )
   {
      case kEventClassHIObject:
            switch ( GetEventKind( inEvent ) )
            {
                  case kEventHIObjectConstruct:
                        …
                        break;

            }
            break;
```

Construction Handler

```
case kEventHIObjectConstruct:
    {
        HIObjectRef         objectRef;

        GetEventParameter( inEvent,
            kEventParamHIObjectInstance,
            typeHIObjectRef, NULL,
            sizeof( HIObjectRef ), NULL,
            &objectRef );

        err = ConstructMyObject( objectRef,
                        &myObjPtr );
```

Construction Handler

```
    err = ConstructMyObject( objectRef,
            &myObjPtr );

    if ( err == noErr )
    {
        // Replace the instance! This is
        // required! Must use typeVoidPtr.

        SetEventParameter( inEvent,
            kEventParamHIObjectInstance,
            typeVoidPtr, sizeof( void * ),
            &myObjPtr );
    }
}
```

You're Done . . Almost . . .

- After the construction phase, your object is now ready to receive events
  - The event handler you specified when you registered your subclass is all set up
- But there is one more phase to take care of . . .

Initialization

- After successful construction, your object is sent an initialization event
  - Passed to HIObjectCreate
- This is how you receive creation parameters
- You should always call through to superclass first
  - CallNextEventHandler
  - Look at error that comes back

Handling the Initialize Event

```
case kEventHIObjectInitialize:
  {
        MyObjectPtr* objPtr = (MyObjectPtr*)inUserData;

        result = CallNextEventHandler( inCallRef, inEvent );
        if ( result == noErr )
            result = objPtr->Initialize( inEvent );
  }
  break;
```

You Have Been Assimilated

- After the initialization phase, your object is fully constructed
- You may now play in the world of HIObject!

Debugging Support

- HIObjectPrintDebugInfo
  - Prints debugging information to stdout
- Your object can add its information
  - kEventHIObjectPrintDebugInfo

More Polymorphism

- HIObjectCopyClassID
- HIObjectIsOfClass
- HIObjectGetEventTarget

Collections and Retain/Release

- How do you put an HIObject into a collection?
- How do you retain or release HIObjects?

Core Foundation!

- HIObjectRefs are actually CF types
- You can retain/release them with CFRetain/CFRelease
- You can add them to dictionaries, arrays, etc.

All Good Things . . .

- When refcount goes to zero, it's time to leave
- Destruction is done top-down
    - Natural event flow
- Do not call through (unless you like to crash)

Handling the Destruct Event

```
case kEventHIObjectDestruct:
  {
        MyObjectPtr* objPtr = (MyObjectPtr*)inUserData;

        delete objPtr;

        result = noErr;
  }
  break;
```

Demo

Summary

- We have been working for several years towards a very exciting future!
- HIObject is that future
  - We've now completed our object model
- Start to use HIObject and our new controls
  - We need your feedback!

Who to Contact

Xavier Legros
Mac OS X Evangelist
xavier@apple.com

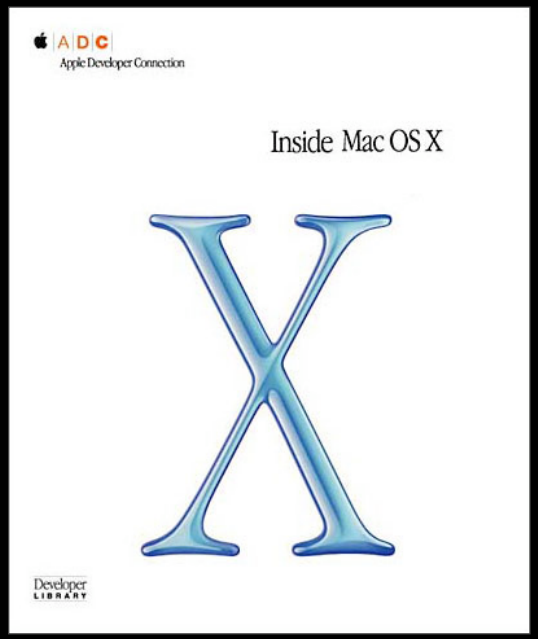
http://developer.apple.com/wwdc2002/urls.html

Roadmap

203 Migrating to Carbon Events	Hall 2 **Tue., 5:00pm**
205 HIToolbox: Introducing HIView	Hall 2 **Wed., 10:30am**
206 HIToolbox: New Controls and Services	Hall 2 **Wed., 2:00pm**
207 Improving Performance with Carbon Events	Hall 2 **Wed., 3:30pm**

Documentation

HIObject



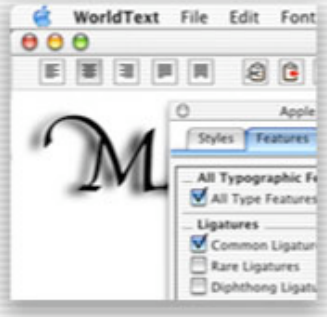
- HIView Reference (Prelim)
- HIObject Reference (Prelim)
- HIToolbar Reference (Prelim)

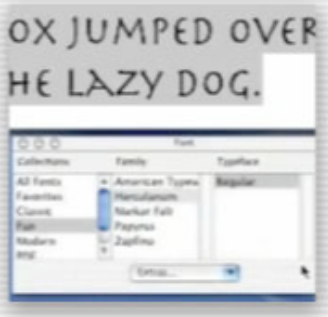
ADC Member Site > Download Software > “Jaguar” Mac OS X

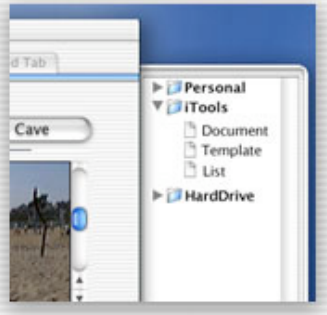
For More Information

- O'Reilly "Learning Carbon"
- Carbon Developer Documentation
  http://developer.apple.com/techpubs/macosx/macosx.html

Q&A







Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002